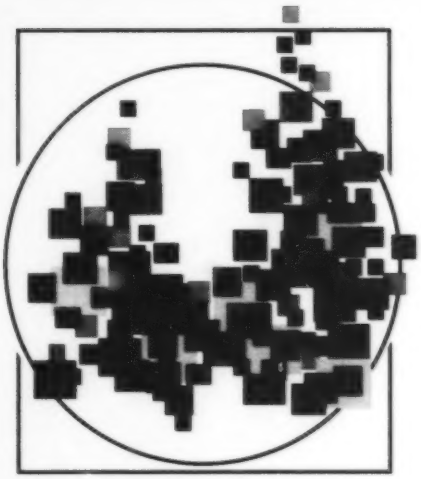


CENTRE FOR
SYSTEMS
SCIENCE

Simon Fraser University, Burnaby, B.C., Canada V5A 1S6 / (604) 291-4704

92-05406/

# Dynamic Programming in Hough Space for Line Matching in Stereopsis

Ze-Nian Li

CSS-IS TR 88-04

# Dynamic Programming in Hough Space for Line Matching in Stereopsis

Ze-Nian Li
School of Computing Science
Simon Fraser University
Burnaby, B.C. V5A 1S6

## ABSTRACT

This paper presents a technique of using the Hough space for solving correspondence problem in stereo matching. The technique is proposed for depth recovery in our robot vision system which will work for a mobile robot in an office enviornment. It is shown that the line matching problem in the image space can be readily converted into a point matching problem in the Hough ($\rho$–$\theta$) space. Dynamic programming can be used for searching the optimal matching, now in the Hough space. The computational cost for dynamic programming in the Hough space is $O(MN)$, where $M$ and $N$ are the numbers of the lines in the left and right images for each $\theta$. Our preliminary results show that the proposed method work well on two sets of test images (Rubik cubes and corridor scenes).

**Index Terms:** Correspondence, Depth, Dynamic Programming, Hough Transform, Matching, Stereo

## 1. Introduction

The recovery of depth information is crucial in three-dimensional image analysis. One method for depth recovery is binocular stereopsis, in which a pair of images from slightly different perspectives are used. One of the most important and difficult problems in stereo vision is *correspondence*. Once corresponding points in the two images are identified, their *disparity* values can be calculated and used to determine the depth.

In 1976 Marr and Poggio [1] demonstrated a cooperative algorithm for the computation of stereo disparity. Two constraints are used. The first is the *uniqueness* constraint which states that each item from each image may be assigned at most one disparity value. The second is the *continuity* constraint which emphasizes that disparity varies smoothly almost everywhere. (Marr and Poggio also said that "... a small fraction of the area of an image is composed of boundaries that are discontinuous in depth.") According to these two constraints the algorithm uses both excitatory and inhibitory inputs to each node. The process is executed iteratively until it converges to a stable state. Remarkable results are shown from the matching of random-dot stereograms.

The constraint of *figural continuity* is proposed by Mayhew and Frisby in 1981 [2]. They stated, "... the edges of surfaces and surface markings such as lines and blobs will be spatially continuous and that is the ultimate justification for relying on figural grouping rules to guide binocular combination." In other words, disparities are continuous along the edges of the surfaces. Since then, this has been accepted as a more powerful disparity-continuous constraint. In his 1985 paper [3], Grimson presented a version of Marr-Poggio-Grimson algorithm that embodied the modifications and improvements to the original Marr-Poggio algorithm. The algorithm used zero-crossings as its basic features. Figural continuity was used to distinguish correct matches from random feature-point matches.

Baker and Binford [4], Ohta and Kanade [5] presented interesting stereo matching algorithms using *dynamic programming*. In [5] the program applies both intra- and inter-scanline searches. The correspondence problem is cast as a problem of finding the optimal path in 2-D and 3-D search planes. Edge-delimited intervals are used as elements to be matched. In the intra-scanline search the cost of a primitive path is defined as the similarity between intervals delimited by edges in the right and left images on the same scanline. In the inter-scanline search, the figural continuity is employed to check the consistency among the scanlines. The dynamic programming technique is shown to be efficient for both searches. Recently, Ohta et al. [6] described a high-speed hardware which is capable of conducting dynamic programming for their stereo matching.

During the past decade, more and more feature-based stereo matching algorithms have been developed. From the literatures, the selection of the size of the matching entities had a complete "spectrum". At one end of the spectrum, matching has been performed at the pixel level. It is basically an area-correlation process. This could be expensive and sensitive to variations in the pixel properties. It is realized that the correspondence problem is too difficult if the matching entities are merely pixels or simple edge elements, since the number of the matching entities is too large and the useful pixel properties are minimal. Nevertheless, the random-dot stereogram does demonstrate that high-level descriptions are not a necessary requirement for stereopsis. At the other end, matching has been performed at very high levels. For instance, Lim and Binford [7] recently suggested a system that executed top-down matching from the levels of body, surface, down to the levels of junction, curve, and edgel. In author's opinion this would not be appropriate. After all, the recognition of 3-D bodies, surfaces is excessively difficult before the depth recovery. In their work [8, 9], Medioni and Nevatia used linear features. It was pointed out that matching at the level of object boundaries seemed to be a suitable compromise. One of the major advantages of line matching techniques is that they naturally incorporate the constraint of figural continuity. The process for matching linear features is still quite complex. This is

somewhat shown from the segment-based matching algorithm in [9]. Lately, McIntosh and Mutch [10] developed a system for matching straight lines. Multiple (eight) features are used. Lines fail to match only when their three or more features differ significantly between images. The matching technique is applied to image registration, and stereo and motion analysis.

We are developing a vision system for a mobile robot (in this case, actually a bipedal walking one). The proposed domain is an office environment. As a first step, the system will use stereo to recover depth information, and then figure out the free walking space. Since most of the obstacles will be man-made objects, it appears appropriate that lines and curves are chosen as the matching entities [11].

The Hough transform [12] has been known to be powerful in extracting lines and curves. In this process lines (curves) in image space are transformed into peak points in the parameter space, or Hough space. It is discovered that the line (curve) matching problem in the image space can be readily converted into a peak point matching problem in the Hough space. Dynamic programming can be done in the Hough space, namely, the matching entities are the peak points in the Hough space. It turns out that if the lines are represented by the parameters $\rho$ and $\theta$, then the search process for the dynamic programming can be independently executed for each $\theta$, and is therefore very efficient. After the correct matching between corresponding lines is made, it is not difficult to obtain the disparity values of the corresponding points on the lines.

This paper presents a technique of using Hough space for solving the correspondence problem in stereo matching. Line matching is the part that has been implemented. Section 2 describes the algorithm for line matching in the Hough space using dynamic programming. Section 3 presents our experimental results. Section 4 concludes this paper.

## 2. Hough Method for Correspondence

This section describes the techniques of line matching and dynamic programming in the Hough

space.

### 2.1. Line Matching in Hough Space

Let $L_l$ and $L_r$ be the corresponding lines in the left and right images, they can be represented by

$$L_l: \quad \rho_l = x_l \cos\theta_l + y_l \sin\theta_l$$

$$L_r: \quad \rho_r = x_r \cos\theta_r + y_r \sin\theta_r$$

After the Hough transform [12], lines can also be represented in their parameter ($\rho$–$\theta$) space (Hough space). For stereo images, two ($\rho$–$\theta$) accumulator arrays can be used for the left and right images, respectively. After the points on $L_l$ and $L_r$ vote on the accumulators, peaks will be formed at $(\rho_l, \theta_l)$ in the left Hough space, and at $(\rho_r, \theta_r)$ in the right Hough space, each representing a line. Because of the *epipolar* constraint, each pair of the matching points on the corresponding lines in the left and right images are considered to have an identical $y$ coordinate, i.e., $y_l = y_r$. Moreover, if the displacement $b$ of the two cameras is relatively small, then it can be assumed that $\theta_l \approx \theta_r$. Accordingly,

$$(\rho_l - \rho_r) \approx (x_l - x_r)\cos\theta, \tag{1}$$

where $\theta$ is approximately equal to $\theta_l$ and $\theta_r$.

Therefore, the following line matching algorithm for deriving disparity values can be suggested.

**Algorithm:**

(1) For each peak in the Hough space, maintain a list of the coordinates for the points that voted for this peak.

(2) Find the corresponding peaks representing $L_l$ and $L_r$ in the left and right Hough spaces. The search space for matching peaks in the Hough spaces is quite limited , because

(a) For each peak with $\theta_l$ in the left accumulator, only peaks with $\theta_l \pm \Delta\theta$ are possible matching candidates in the right accumulator, while $\Delta\theta$ is a small number. (In this paper we only discuss matching from left to right. Although matching from right to left can be

added in an obvious way.)

(b) If the estimated disparity ($d = x_l - x_r$) values are in the range of $[d_1, d_2]$, then the displacement ($\rho_l - \rho_r$) will be in the range of $[d_1 \cos\theta,\ d_2 \cos\theta]$. Thus, only peaks with $\rho_r \in [\rho_l - d_1 \cos\theta,\ \rho_l - d_2 \cos\theta]$ are possible matching candidates in the right accumulator array.

(3) Assign disparity values to individual points on the matched lines $L_l$ and $L_r$. Points that have an identical $y$ coordinate are candidate matching points. Their $x_l - x_r$ are taken as disparity ($d$) values. Fig. 1 illustrates a possible point matching situation after the two lines have been chosen as corresponding pairs. One of the advantages of using Hough transform is that it is tolerant to noisy and broken lines. In case a slightly broken line is caused by a partial occlusion in one image but not in the other, two corresponding peaks will still be formed and matched. As shown in Fig. 1, a reasonable match between most other points is still possible.

Fig. 1 Point Matching after Line Matching

It is likely that the disparity $d$ assigned to each individual point is slightly different from the estimated disparity for the whole line. In other words, the disparity for each individual point is refined to a higher accuracy.

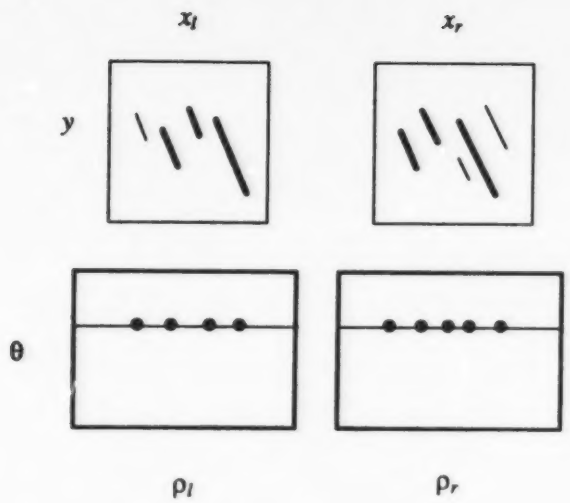
## 2.2. Dynamic Programming in Hough Space

By choosing lines as matching entities, the amount of data for correspondence is significantly reduced. This section shows the implementation of step 2 in the proposed algorithm, i.e., dynamic programming in the Hough space. For the simplicity of this description it is temporarily assumed that $\theta_l = \theta_r$.

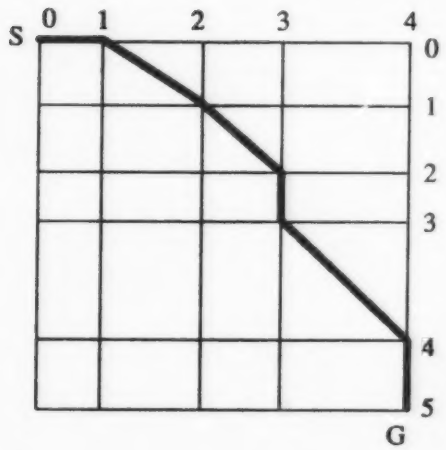
As depicted in Fig. 2(a), for a particular $\theta$ there may exist more than one contending lines in both left and right images. Accordingly, more than one peaks are located for the same $\theta$ in both left and right Hough spaces. Often, no one-to-one correspondence exists between the lines appeared in the left and right images. This is because (a) the leftmost lines in the left image may be shifted out of the image plane in the right image, and vice versa; (b) the occlusions of lines may be different from the left and the right view; (c) the possible difference between the noise, reflectance in the two images. Therefore, for each $\theta$, it is necessary to find the correspondence between $N$ peak points in the left Hough space and $M$ peak points in the right Hough space.

Fig. 2(b) illustrates a search plane for dynamic programming in the Hough space. The candidate matching lines are the ones in Fig. 2(a), they are all oriented at a particular $\theta$. The horizontal axis shows the peaks in the left Hough space and the vertical axis shows the peaks in the right Hough space. Along each $\theta$ in the Hough space, peaks are scaned from left to right. In this example, $N = 4$ and $M = 5$.

As suggested by Burt and Julesz [13], binocular fusion cannot be obtained when the *disparity gradient* exceeds a certain limit. The disparity gradient limit implies the constraint of *nonreversal order*, i.e., the left-to-right order of objects cannot be reversed in one eye with respect to the other. In our case, this implies that the orders of the matching peaks in the Hough space should not be reverseable either.




(a)




(b)

**Fig. 2 Dynamic Programming for Line Matching in Hough Space. (a) Lines in the image ($x$–$y$) space and their corresponding peak points in the Hough (ρ–θ) space; (b) The search plane for matching lines in (a).**

### 2.2.1. Optimal Solution Path

The process of dynamic programming in the Hough space is similar to the dynamic programming in the image space described by Ohta and Kanade [5]. Instead of matching features in the image space, peaks in the Hough space (representing lines in the image space) are matched. The correpondence problem is now a problem of finding an *optimal solution path* (with the least cost) on the search plane. Nodes in the search plane correspond to peaks in the Hough space. A solution path is any path from $S$ to $G$ (from (0, 0) to $(M, N)$). The horizontal segment from $(i, j-1)$ to $(i, j)$ on a path corresponds to a no-match for line $j$ in the left image [e.g., the dark horizontal segment from (0, 0) to (0, 1) in Fig. 2(b) indicates that the first (leftmost) line in the left image has no matching line in the right image]; the vertical segment from $(i-1, j)$ to $(i, j)$ on a path corresponds to a no-match for line $i$ in the right image; whereas a diagonal segment from $(i-1, j-1)$ to $(i, j)$ on a path corresponds to a match between line $j$ in the left image and line $i$ in the right image [e.g., the first dark diagonal segment from (0, 1) to (1, 2) in Fig. 2(b) represents the match between the second leftmost line in the left image and the first (leftmost) line in the right image]. Each of the above three types of segments is a *primitive path* and is assigned a cost. Because of the nonreversal ordering constraint, starting from $S$, a path can be extended towards only one of the three directions: east, south, or southeast.

Similar to [5], let $C(\mathbf{m})$ be the cost of the optimal path from the origin (0, 0) to node $\mathbf{m}$, $c(\mathbf{m}, \mathbf{m}-\mathbf{i})$ be the cost of the primitive path from $\mathbf{m}-\mathbf{i}$ to $\mathbf{m}$, then the cost of all the nodes in the search plane can be recursively defined as

$$C(\mathbf{o}) = 0,$$

$$C(\mathbf{m}) = \min_{\{\mathbf{i}\}} [c(\mathbf{m}, \mathbf{m}-\mathbf{i}) + C(\mathbf{m}-\mathbf{i})],$$

$$\text{where } \mathbf{o} = (0, 0);\ \mathbf{i} = (i, j),\ 0 \le i \le 1, 0 \le j \le 1,\ \text{and } i + j \ne 0.$$

The path that renders $C(\mathbf{G})$ at node $\mathbf{G} = (M, N)$ is the optimal solution path.

It is presumed that the search process is Markovian, namely, $C(\mathbf{m})$ only depends on its immediate predecessor's $C(\mathbf{m}-\mathbf{i})$ and the $c(\mathbf{m}, \mathbf{m}-\mathbf{i})$. It does not depend on the previous history, i.e., the way of deriving $C(\mathbf{m}-\mathbf{i})$.

### 2.2.2. Computational Cost for the Search

Since each node $\mathbf{m}$ has only three possible predecessors $\mathbf{m}-\mathbf{i}$, i.e., its immediately west, north, or northwest neighboring node, the cost computation for all the nodes can be done in one scan (row-by-row) through the 2D search plane, starting from $S$ and ending at $G$.

The number of nodes in the search plane is $O(MN)$. At each node we only examine three primitive paths, denoted as $I = 3$. Because $\theta_l$ is only approximately equal to $\theta_r$, it is suggested in step 2 of the algorithm that search be conducted at the range of $\theta_l \pm \Delta\theta$ in the right accumulator. If we decided to examine two more angles, e.g., $\theta_l + 10^o$ and $\theta_l - 10^o$, then the number of $\theta$'s examined is $J = 3$. Moreover, it is also suggested in step 2 of the algorithm that only peaks with $\rho_r \in [\rho_l - d_1 \cos\theta,\ \rho_l - d_2 \cos\theta]$ are possible matching candidates in the right accumulator array. That means only a portion $k$ of the nodes are to be searched ($k < 1$). Put them all together, the number of primitive paths to be examined is $k \times M \times N \times I \times J$. It follows that the computational cost is $O(MN)$.

Remember, the search is performed on each $\theta$. There are, for example, 36 possibly different $\theta$'s. $M$ and $N$ are the number of the lines in the left and right images for each $\theta$. Therefore, $M$ and $N$ are usually fairly small numbers. This makes the search very fast.

### 2.2.3. Cost Functions

The cost for each segment on the search plane is assigned by a *cost function*. Cost functions are heuristic functions. The following heuristics seem to be highly feasible and are used in this paper:

(1) **Maximize the number of points matched.** The belief is that, usually, less points can be put in pairs if erroneous correspondence is made among the contending lines. Since the horizontal and vertical segments on a path in the search plane imply a no-match, their cost is set to be equal to the length of the line being skipped. This could be costly, in other words, the skipping of long lines is discouraged. The diagonal segment represents a match between a pair of lines. As shown in Fig. 1 there might not be a perfect point-to-point match between the designated matching lines. Let $l_1$ and $l_2$ be the lengths of the lines $L_l$ and $L_r$, $m$ be the number of the points matched on each line, the cost assigned to the diagonal line segment in the search space is $l_1+l_2-2m$.

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 — | 3 | 14 | 27 | 45 | 58 | 66 |
| 12 | 15 | 4 | 17 | 35 | 48 | 56 |
| 26 | 29 | 18 | 5 | 23 | 36 | 44 |
| 47 | 50 | 39 | 26 | 10 | 23 | 31 |
| 59 | 62 | 51 | 38 | 22 | 15 | 23 |
| 67 | 70 | 59 | 46 | 30 | 23 | 19 |
| 74 | 77 | 66 | 53 | 37 | 30 | 26 |

Fig. 3 An Optimal Path That Maximizes the Number of Points Matched

Fig. 3 shows an example of an optimal path generated by our program with a cost function that maximizes the number of points matched between 6 left lines and 6 right lines. These line all have approximately the same θ. The lengths for the 6 left lines are 3, 11, 13, 18, 13, and 8. The lengths for the right lines are 12, 14, 21, 12, 8, and 7. The number in (0, $j$) indicates the total cost for skipping the first $j$ left lines. The number in ($i$, 0) indicates the total cost for skipping the first $i$ right lines. In general, all the numbers indicate the optimal cost $C$ for getting to the node. Let us take node (4, 5) as an example to see the calculation of $C$ (4, 5). As said before,

there are three paths to reach the node (4, 5) and their cost are

$$c\ ((4,5),(4,4)) + C\ (4,4) = 13+22 = 35,$$

$$c\ ((4,5),(3,5)) + C\ (3,5) = 12+23 = 35,$$

$$c\ ((4,5),(3,4)) + C\ (3,4) = 5+10 = 15.$$

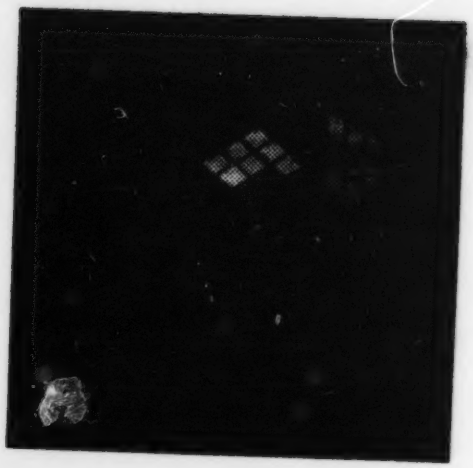
Apparently, the last has the least cost 15. Therefore, $C\ (4,5) = 15$. [ $c\ ((4,5),(3,4))$ indicates that there is a partial match between the fifth left line and the fourth right line. 10 points from each line are matched. The cost for this diagonal segment is $l_1+l_2-2m = 13+12-2\times 10 = 5$. ]

It should be pointed out that this heuristic turns out to be very powerful. It is also very efficient to implement, because the length of each line is already stored in each peak node in the Hough space.

(2) **Compare the local property at the vicinity of the lines**. For example, edge magnitude can be used to measure the difference between the gray level intensities at the two sides of an edge. It can be enforced that two correctly matched lines (edges) should have similar edge magnitude. Higher cost can be assigned to the matching pair that has larger differences in the local properties.

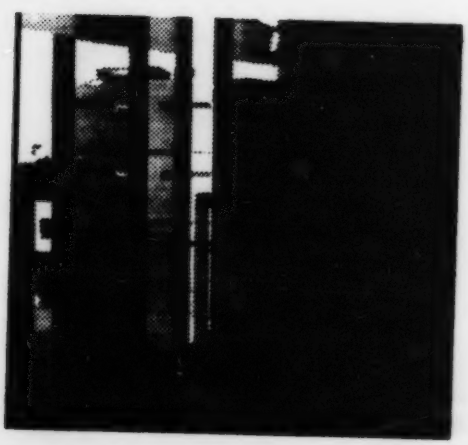
### 3. Experimental Results

This section presents some preliminary experimental results. In our system two cameras are used. They are mounted on the same horizontal level and separated with a small distance $b = 4$ inches. The first set of the stereo data are two Rubik cubes (Fig. 4 (a), (b)). They are frequently used in the stereo vision literatures, because the repetitive patterns often cause some difficulty for the correspondence process. The second stereo pair is an ordinary indoor scene (Fig. 4 (c), (d)). It is a corridor scene, typical for our robotics environment. The first room is an executive office with sliding glass doors and windows. A cart is placed there to add a little more complexity.

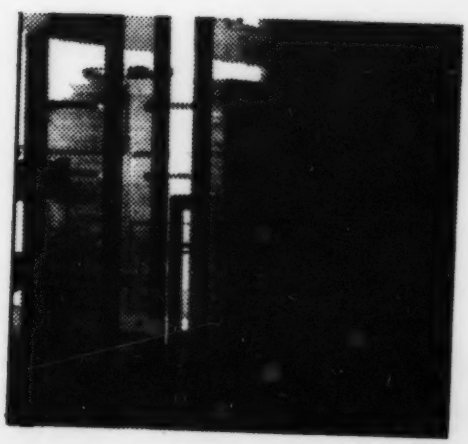


(a) Left Cube

(b) Right Cube



(c) Left Corridor



(d) Right Corridor

Fig. 4 Test Stereo Images

### 3.1. Edge Detection and Pyramidal Linking

Many edge extraction algorithms use a set of edge masks. For example, a standard set of Prewitt gradient operators includes eight edge masks. The edge direction θ is thus approximated as one of the eight possible angles (0°, 45°, ... , 315°). For representing lines in the Hough space more accuracy in θ is desirable. The following is recommended by Rosenfeld, Ornelas, and Hung [14]:

"At pixel $P$ whose neighbors have gray levels $\begin{matrix} A & B & C \\ D & P & E \\ F & G & H \end{matrix}$ we first compute the Sobel differences

$$\Delta_x = \frac{1}{4}\,[(C+2E+H)-(A+2D+F)]$$

$$\Delta_y = \frac{1}{4}\,[(A+2B+C)-(F+2G+H)].$$

The magnitude and direction are then defined as

$$m = \sqrt{\Delta_x^2+\Delta_y^2} \quad \text{and} \quad \theta = \tan^{-1}(\Delta_y / \Delta_x).$$

respectively. We call $P$ an edge pixel if the value of $m$ at $P$ exceeds some specified threshold."

The edges thus extracted are usually more than one pixel wide. The maximum-suppression technique is used to reduce their edge widths to one. A pixel will survive as an edge pixel if its edge magnitude is greater than all of its eight neighbors'.

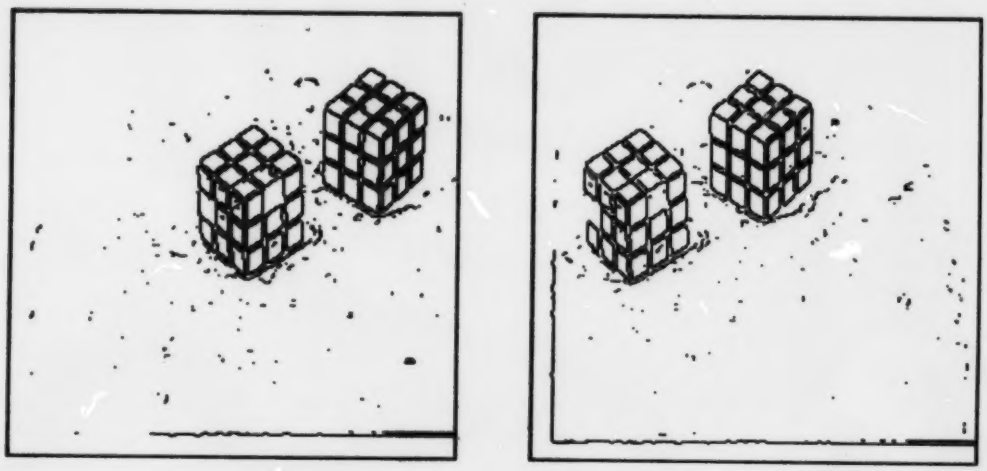
The original images have fairly high resolutions (512×512 for the cubes, 512×484 for the indoor scenes). To save computation time it is decided that line matching will be performed on images with half of their original resolutions, e.g., 256×256 for cubes. To improve the quality of edge detection at the reduced resolution, especially to improve the accuracy of the edge directions, *pyramid linking* [15] is applied.

A pyramid has $n$ levels. Each level k has $2^k \times 2^k$ nodes. For the purpose of linking, an overlapped pyramid is used in which each parent at level $k$ can have links to its 4×4 children at level $k+1$. Consequently, a child can have up to four parents.

Initially, the input image at level 9 is reduced to a resolution of 256×256 at level 8. Edges are extracted at both level 9 and 8. A child is linked to one of its four parents that has the closest

edge direction. After this initial linking, the edge direction for the parent node is adjusted. The average edge direction of its linked children is now taken as the parent's edge direction. This process is based on the observation that edges are usually continuous and their directions change smoothly. The pyramid linking process can be applied iteratively. In this experiment only one iteration is applied.

Fig. 5 are the edge maps for the test images at level 8.



(a) Left Cube (b) Right Cube

(c) Left Corridor (d) Right Corridor

Fig. 5 Edge Maps for Test Images at Level 8

### 3.2. Implementing the Hough Transform

As suggested by Duda and Hart [12], a $\rho$–$\theta$ accumulator array is used as the Hough space for the lines. The divisions are every 10° for $\theta$ and every 2 pixels for $\rho$. Since the gradient information is used, an edge point $e_0$ will vote for only one corresponding point $(\rho_0, \theta_0)$ in the accumulator array. After all the edge points in the image space have voted, local maximum points (peaks) are extracted to represent the lines.

Differed from the original paper [12], the range of our accumulator array is: $0° \le \theta \le 360°$, $-\rho_{max} \le \rho \le \rho_{max}$. This doubles the Hough space from the original paper. Fig. 6 illustrates edges caused by intensity changes between dark regions and bright regions (in this paper, "edge" and "line" are used interchangeably). The direction of $\theta$ is perpendicular to the edge, pointing to the bright side. Fig. 5 shows some examples of lines with positive or negative $\rho$'s. In this way, different edges are more carefully represented, and they can be readily distinguished in the Hough space.

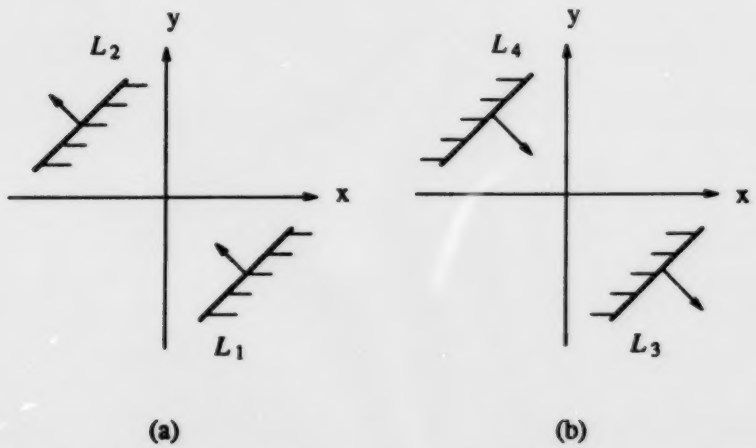



Fig. 6 Lines with Positive or Negative $\rho$'s (a) $L_1$: $\theta_1 = 135°$, $\rho_1 < 0$; $L_2$: $\theta_2 = 135°$, $\rho_2 > 0$. (b) $L_3$: $\theta_3 = 315°$, $\rho_3 > 0$; $L_4$: $\theta_4 = 315°$, $\rho_4 < 0$.

As described in Section 2.1, it is necessary to keep the coordinate information for all the voting edges. Hence, a linked list is maintained at each cell in the accumulator array. For easy

access, the list is sorted by their $x, y$ values. The sorting turns out to have no overhead, since the edge points were examined in order at the time when their votes were taken.

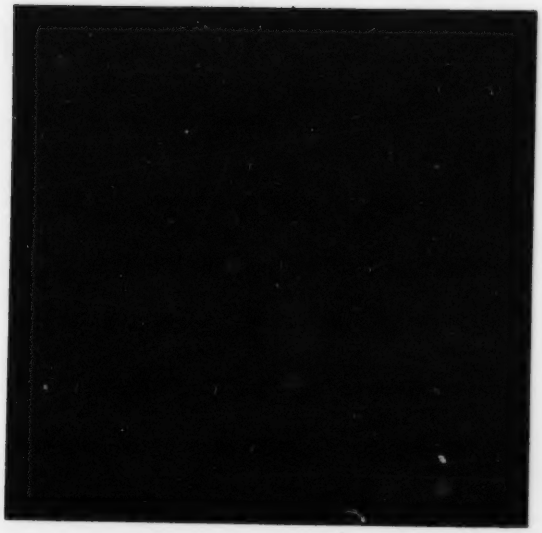
### 3.3. Dynamic Programming for Matching

The search process of dynamic programming has been discussed in detail in Section 2.2. For the cost function, a combination of the two criteria is used. In our experiment, emphasis has been placed on the first criterion, i.e., maximizing the number of points that can be matched.

When we described the concept of matching in the Hough space, we said that peak points in the Hough space represented lines in the image space, and they were eventually put in matching pairs. In the real implementation, *clusters* are used in place of the peak points. A cluster is a small rectangular area in the accumulator array centered at the location of the peak point. Because of the inevitable error of digitization and especially the inaccuracy in the computation of the direction $\theta$, a peak in the Hough space is never as sharp as a single point. It is more likely that all the points on a line in the image will map into a cluster with a reasonable size in the Hough space. In our implementation, if the peak point is located at $(\rho_0, \theta_0)$, then a cluster will typically have $\rho_0 - 3 \le \rho \le \rho_0 + 3$ and $\theta_0 - 10^\circ \le \theta \le \theta_0 + 10^\circ$.

### 3.4. The Depth Maps

Fig. 7 are the depth maps generated by our program. In binocular stereo vision, depth is inversely proportional to the disparity. For the purpose of display in black and white outputs, depths are shown by various gray level intensities. The closer points are shown brighter, the more distant points are dimmer.



(a) Cube

(b) Corridor Scene

Fig. 7 Depth Maps for Test Images

The accuracy of the depth is determined by the resolution of the matching stereo images if the correct correspondences are made. For example, for our cube images, the distance between

the camera and the closest corner of the front cube is $z_1 = 93.60$ inches, the distance between the camera and the farthest corner of the back cube is $z_2 = 98.75$ inches. The corresponding disparity values obtained by our program are $d_1 = 81$ and $d_2 = 77$ using the $256 \times 256$ images. Ideally, $d_1 \times z_1 = d_2 \times z_2$. It can be seen that the relative error is approximately 0.3%. Indeed, the error for disparity is at the order of $\frac{1}{256}$.

The two cameras in our system are placed on the same horizontal line with a displacement $b$. It is clear that this configuration is not suitable for the stereo matching between horizontal lines, unless the end points of the horizontal lines can be located. In our images of corridor scenes, many horizontal lines (e.g., the file cabinets) are occluded by some other objects, the end points of these lines cannot be reliably located. Hence, matching is not even attemped for these lines. The outputs for horizontal (or nearly horizontal lines) are not indications of their depth. Rather, they just indicate the existence of these edges for the ease of viewing. A quick solution for this might be using three cameras as suggested by some researchers [16].

Mohan et al. [17] have suggested methods for detecting matching errors. Subjectively, we can also examine the disparity map and assign the error rates. If we count the matched point that has a disparity deviated from the correct value by more than 2% as an error, then the error rate is 6.6% for the cube images and 4.8% for the corridor images. The disparity errors mostly occur on the slanted edges in the cubes, and on the noisy (textured) floor points in the corridors.

Our program runs on a Sun-3 workstation. The edge detection and linking process is very slow. It runs on the order of 10 minutes. The Hough transform part takes 65 seconds of cpu time. The dynamic programming only takes 11.2 seconds of cpu time.

## 4. Conclusion

This paper presented a technique of using the Hough space for solving correspondence problem in stereo matching. The technique is proposed for depth recovery in our robot vision system

which will work for a mobile robot in an office enviornment. Since most of the obstacles are man-made objects, it is appropriate to use well-defined lines and curves as matching entities.

It is shown that the line matching problem in the image space can be readily converted into a point matching problem in the Hough space. The popular method of dynamic programming can be used for stereo matching, now in the Hough space. Cost functions are defined according to two heuristics: (1) The maximization of the number of points matched, (2) The similarity of the local property at the vicinity of the lines. A search plane consists of nodes that correspond to peak points in the Hough space. Each line segment on the search plane is assigned a cost by the cost function. The correspondence problem is now a problem of finding an optimal solution path on the search plane. The dynamic programming determines the correspondence between lines. The computational cost for the search of dynamic programming in the Hough space is $O(MN)$, where $M$ and $N$ are the numbers of the lines in the left and right images for each $\theta$. Since $M$ and $N$ are usually fairly small numbers, the search is very efficient. After the line matching, individual points on these lines are paired and given their disparity values.

Our preliminary results show that the proposed method work well on two sets of test images (Rubik cubes and corridor scenes).

The generalization of this technique is necessary, especially in terms of matching 3-D curves. The obvious difficulty is that the shape of the 3-D curves would change from different perspectives. In the line matching, the shape of the lines never change. The slight perspective difference in their orientations is taken care of by examining a few more nodes in the $(\rho-\theta)$ space with $\pm\Delta\theta$.

## REFERENCES

[1] D. Marr and T. Poggio, "Cooperative computation of stereo disparity," *Science* **194** pp. 283-287 (Oct. 15, 1976).

[2] J.E.W. Mayhew and J.P. Frisby, "Psychophysical and computational studies towards a theory of human stereopsis," *Artificial Intelligence* **17** pp. 349-385 (1981).

[3] W.E.L. Grimson, "Computational experiments with a feature based stereo algorithm," *IEEE Trans. on Pattern Analysis and Machine Intelligence* **PAMI-7**(1) pp. 17-34 (1985).

[4] H.H. Baker and T.O. Binford, "Depth from edge and intensity based stereo," *Proc. 7th IJCAI*, pp. 631-636 (1981).

[5] Y. Ohta and T. Kanade, "Stereo by intra- and inter-scanline search using dynamic programming," *IEEE Trans. on Pattern Analysis and Machine Intelligence* **PAMI-7**(2) pp. 139-154 (1985).

[6] Y. Ohta, K. Takano, and K. Ikeda, "A highspeed stereo matching system based on dynamic programming," *Proc. 1st Int. Conf. on Computer Vision*, pp. 335-342 (1987).

[7] H.S. Lim and T.O. Binford, "Structural correspondence in stereo vision ," *Proc. DARPA Image Understanding Workshop*, pp. 794-808 (1988).

[8] G. Medioni and R. Nevatia, "Matching images using linear features," *IEEE Trans. on Pattern Analysis and Machine Intelligence* **PAMI-6** pp. 675-685 (1984).

[9] G. Medioni and R. Nevatia, "Segment-based stereo matching," *Computer Vision, Graphics, and Image Processing* **31** pp. 2-18 (1985).

[10] J.H. McIntosh and K.M. Mutch, "Matchng straight lines," *Computer Vision, Graphics, and Image Processing* **43** pp. 386-408 (1988).

[11] A. de Saint Vincent, "A 3D perception system for the mobile robot HILARE," *Proc. IEEE Int. Conf. on Robotics and Automation*, pp. 1105-1111 (1986).

[12] R.O. Duda and P.E. Hart, "Use of the Hough transform to detect lines and curves in pictures," *Communications of the ACM* **15** pp. 11-15 (1972).

[13] P. Burt and B. Julesz, "Modifications of the classical notion of Panum's fusional area," *Perception* **9** pp. 671-682 (1980).

[14] A. Rosenfeld, J. Ornelas, and Y. Hung, "Hough transform algorithms for mesh-connected SIMD parallel processors," *Computer Vision, Graphics, and Image Processing* **41** pp. 293-305 (1988).

[15] P.J. Burt, "The pyramid as a structure for efficient computation," pp. 6-35 in *Multiresolution Image Processing and Analysis*, ed. A. Rosenfeld, Springer-Verlag (1984).

[16] M. Pietikainen and D. Harwood, "Depth from three camera stereo," *Proc. IEEE Conf. on Computer Vision and Pattern Recognition* , pp. 2-8 (1986).

[17] R. Mohan, G. Medioni, and R. Nevatia, "Stereo error detection, correction and evaluation," *Proc. 1st Int. Conf. on Computer Vision*, pp. 315-324 (1987).

## Centre for Systems Science

---

• Simon Fraser University established the **Centre for Systems Science** as a provincial centre for advanced study. The objective of CSS is twofold: coordinate and combine advanced systems research at SFU, and encourage application of research results within industry in order to further the technological development of British Columbia.

• CSS has assembled three research groups to streamline its efforts. The orientation of these groups is as follows: intelligent systems, microelectronics, and computer and communications systems. Because of shared research interests, this latter group merges with LCCR (**Laboratory for Computer and Communications Research**) for the publication of technical reports.

• CSS's three groups involve over 80 skilled faculty researchers principally from SFU's Schools of Engineering Science and Computing Science, but also draw substantially on Mathematics, Physics, Geography, Philosophy, Linguistics, Psychology, Communication, and Kinesiology.